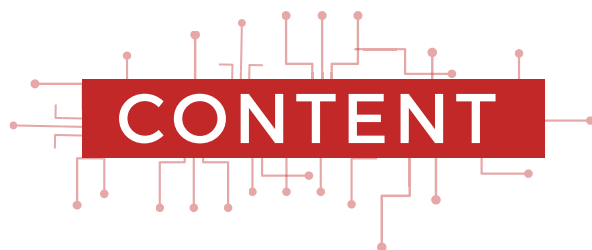




LP Staking AvaXLauncher Smart Contract Final Audit Report

SEP 2021



●	Scope of Audit	2
●	Check Vulnerabilities	2
●	Techniques and Methods	3
	Issue Categories	
	Number of security issues per severity	
●	Introduction	5
●	Issues Found – Code Review / Manual Testing	6
	High Severity Issues	
	Medium Severity Issues	
	Low Severity Issues	
	A.1 getRewards not updated after staking	
	A.2 After unStake rewards got zero, update in global var	
	Informational Issues	
	A.1 Public function that could be declared external	
	A.2 SafeMath not used	
●	Functional Tests	8
●	Unit Tests	9
●	Automated Tests	14
	Slither	
	Results	
	Surya	
	Results	
●	Closing Summary	20
●	Disclaimer	20

A decorative graphic featuring a central red rectangle with the text 'SCOPE OF AUDIT' in white. The rectangle is surrounded by a complex, symmetrical pattern of red lines and dots, resembling a circuit board or a stylized brain.

SCOPE OF AUDIT

The scope of this audit was to analyze and document the AvaXLauncher Lp-Stake smart contract codebase for quality, security, and correctness.

A decorative graphic featuring a central red rectangle with the text 'CHECK VULNERABILITIES' in white. The rectangle is surrounded by a complex, symmetrical pattern of red lines and dots, resembling a circuit board or a stylized brain.

CHECK VULNERABILITIES

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC-20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

TECHNIQUES AND METHODS

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of BEP-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

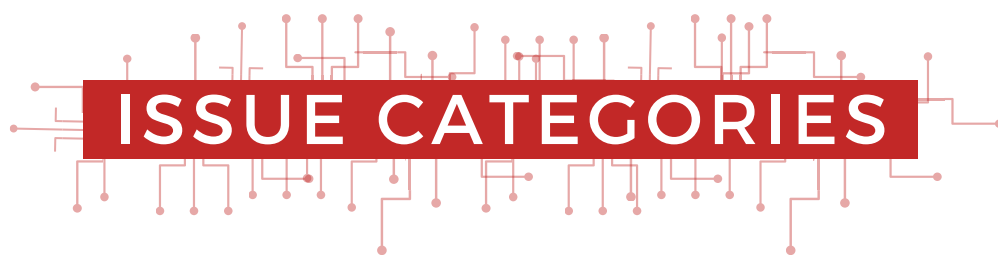
Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.



ISSUE CATEGORIES

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

NUMBER OF SECURITY ISSUES PER SEVERITY

TYPE	HIGH	MEDIUM	LOW	INFORMATIONAL
Open	0	0	2	2
Acknowledged	0	0	2	2
Closed	0	0	2	2

Introduction

During the period of September 14, 2021 to September 20, 2021 - Crypticocean Team performed a security audit for AvaXLauncher smart contracts.

ISSUES FOUND – CODE REVIEW / MANUAL TESTING

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

A.1 GetRewards not updated after unstake

Description

Get rewards struct variables are not getting updated when unstake.

Remediation

Update the variable inside the scope of function

Status: Closed

A.2 After unStake rewards got zero, update in global var

Description

After Unstake, total rewards pending become zero, use global variable to update rewards of a staker after unstaked by user

Remediation

Update global variable for every change in rewards stats of a user.

Status: Closed

Informational Issues

A.1 Public function that could be declared external

Description

The following public functions that are never called by the contract should be declared external to save gas:

- ERC20.totalSupply (StakingContract.sol#335-337) should be declared external
- IERC20.totalSupply (StakingContract.sol#184) should be declared external
- IERC20.balanceOf (StakingContract.sol#189) should be declared external
- ERC20.balanceOf (StakingContract.sol#342-344) should be declared external
- IERC20.transfer (StakingContract.sol#198-200) should be declared external
- ERC20.transfer (StakingContract.sol#354-357) should be declared external
- ERC20.allowance (StakingContract.sol#362-368) should be declared external
- IERC20.allowance (StakingContract.sol#209-212) should be declared external
- ERC20.approve (StakingContract.sol#377-380) should be declared external
- IERC20.approve (StakingContract.sol#228) should be declared external
- ERC20.transferFrom (StakingContract.sol#394-409) should be declared external
- IERC20.transferFrom (StakingContract.sol#239-243) should be declared external
- ERC20.increaseAllowance (StakingContract.sol#423-433) should be declared external
- ERC20.decreaseAllowance (StakingContract.sol#449-462) should be declared external
- Staking.stake (StakingContract.sol#703-733) should be declared external
- Staking.unstake (StakingContract.sol#738-787) should be declared external
- Staking.totalStakers (StakingContract.sol#867-869) should be declared external
- Staking.afterUnstakeStats (StakingContract.sol#985-999) should be declared external
- Staking.getUserStats (StakingContract.sol#1010-1043) should be declared external

Remediation

Use the external attribute for functions that are never called from the contract

Status: Closed

A.2 Reentrancy in unStake

Description

Reentrancy possibilities in unStake function

Remediation

Reentrancy in Staking.unstake (StakingContract.sol#738-787):

External calls:

- claimReward() (StakingContract.sol#756)

State variables written after the call(s):

- rewards (StakingContract.sol#776)

- user (StakingContract.sol#779)

- user (StakingContract.sol#780)

- user (StakingContract.sol#781)

- user (StakingContract.sol#782)

Status: Closed

FUNCTIONAL TESTS

Function Name()	Technical Result	Logical Result	Overall Result
Read Functions()			
getUserStats	Pass	Pass	Pass
totalStaked	Pass	Pass	Pass
totalStakers	Pass	Pass	Pass
afterUnstakeStats	Pass	Pass	Pass
getRewardStats	Pass	Pass	Pass
isStakeHolders	Pass	Pass	Pass
user	Pass	Pass	Pass
Write Functions()			
stake	Pass	Pass	Pass
unStake	Pass	Pass	Pass
claim	Pass	Pass	Pass



UNIT TESTS

- ✓ Should correctly initialize constructor values of AvaxLancher token contract (75ms)
- ✓ Should correctly initialize constructor values of Staking contract (57ms)
- ✓ Should check avxl Contract address in staking contract
- ✓ Should check APY of Staking
- ✓ Should check stake contract user stats
- ✓ Should check stake contract user stats
- ✓ Should check total staked AVXL tokens using function
- ✓ Should check total avxl tokens staked
- ✓ Should check if stake holder or not , when not
- ✓ Should check getUserStats of accounts[1] before staking
- ✓ Should not be able to stake when doesnt have avxl token (70ms)
- ✓ Should not be able to claim tokens before stake (50ms)
- ✓ Should not be able to unstake tokens before stake (55ms)
- ✓ Should check a AVXL balance of a Contract address Staking
- ✓ Should be able to transfer AVXL to staking contract that will be rewarded (52ms)
- ✓ Should be able to transfer AVXL to accounts[2] (103ms)
- ✓ Should check a AVXL balance of a Contract address Staking after (47ms)
- ✓ Should check a AVXL balance of a accounts[1]
- ✓ Should be able to transfer AVXL to staking contract that will be rewarded (43ms)
- ✓ Should check a AVXL balance of a account[1]
- ✓ Should check approval by accounts 1 to Staking contract to spend tokens on the behalf of staking
- ✓ Should Approve staking to spend specific tokens of accounts[1] (72ms)
- ✓ Should check approval by accounts 0 to Staking contract to spend tokens on the behalf of staking after
- ✓ Should be able to stake when have avxl token (63ms)

- ✓ Should check total staked AVXL tokens using function after staked by accounts[1]
- ✓ Should check total avxl tokens staked, after staked by accounts[1]
- ✓ Should check if stake holder or not , after staked by accounts[1]
- ✓ Should check stake contract user stats after staking by accounts[1]
- ✓ Should check a AVXL balance of a Contract address Staking after staked by accounts[1]
- ✓ Should check a AVXL balance of a accounts[1]
- ✓ Should check getUserStats of accounts[1] after staking
- ✓ Should not be able to claim tokens after stake, before vesting of claimable tokens (45ms)
- ✓ Should be able to unstake tokens before stake (67ms)
- ✓ Should check unStake UserData
- ✓ Should not be able to stake when already staked (49ms)
- ✓ Should check total stakers AVXL tokens using function after staked by accounts[1]
- ✓ Should check total avxl tokens staked, after staked by accounts[1]
- ✓ Should check if stake holder or not , after staked by accounts[1]
- ✓ Should check stake contract user stats after staking by accounts[1]
- ✓ Should check a AVXL balance of a Contract address Staking after staked by accounts[1]
- ✓ Should check a AVXL balance of a accounts[1]
- ✓ Should check getUserStats of accounts[1] after staking (44ms)
- ✓ Should not be able to claim tokens after unstake and no claimable tokens (48ms)
- ✓ Should check a AVXL balance of a account[2]
- ✓ Should check approval by accounts 2 to Staking contract to spend tokens on the behalf of staking
- ✓ Should Approve staking to spend specific tokens of accounts[1] (55ms)
- ✓ Should check approval by accounts 0 to Staking contract to spend tokens on the behalf of staking after
- ✓ Should be able to stake when have avxl token (63ms)
- ✓ Should check total staked AVXL tokens using function after staked by accounts[2]

- ✓ Should check total avxl tokens staked, after staked by accounts[2]
- ✓ Should check if stake holder or not , after staked by accounts[2]
- ✓ Should check stake contract user stats
- ✓ Should check stake contract user stats after staking by accounts[2]
- ✓ Should check a AVXL balance of a Contract address Staking after staked by accounts[2]
- ✓ Should check a AVXL balance of a accounts[1]
- ✓ Should check getUserStats of accounts[2] after staking
- ✓ Should not be able to claim tokens after stake, before vesting of claimable tokens (66ms)
- ✓ Should check getUserStats of accounts[4] to know status, when not staked
- ✓ Should be able to increase time to get 37 days
- ✓ Should check Rewards stats of a accounts[2] after 38 days of staking
- ✓ Should be able to claim tokens after stake of 37 days (51ms)
- ✓ Should check a AVXL balance of a Contract address Staking after 1st week, claimed by accounts[2]
- ✓ Should check a AVXL balance of a accounts[2] after claiming 1st week tokens
- ✓ Should check Rewards stats of a accounts[2] after 38 days of staking and after claiming (106ms)
- ✓ Should check stake contract user stats after staking by accounts[2] & after claiming
- ✓ Should not be able to claim tokens after stake, before vesting of claimable tokens (68ms)
- ✓ Should be able to increase time to get 7 days
- ✓ Should check Rewards stats of a accounts[2] after 44 days of staking (53ms)
- ✓ Should be able to increase time to get 7 days
- ✓ Should check Rewards stats of a accounts[2] after 51 days of staking (41ms)
- ✓ Should be able to claim tokens after stake of 51 days (94ms)
- ✓ Should check Rewards stats of a accounts[2] after 51 days of staking (48ms)
- ✓ Should check a AVXL balance of a Contract address Staking after 1st week, claimed by accounts[2]

- ✓ Should check a AVXL balance of a accounts[2]
- ✓ Should not be able to claim tokens after stake, before vesting of claimable tokens (63ms)
- ✓ Should be able to unstake tokens after claiming tokens of 3 weeks and total 7 weeks staking complete (112ms)
- ✓ Should check Rewards stats of a accounts[2] after 51 days of staking and then unstake (38ms)
- ✓ Should be able to increase time to get 100 days
- ✓ Should check Rewards stats of a accounts[2] after 51 days of staking and then unstake
- ✓ Should be able to claim tokens after stake of 51 days (43ms)
- ✓ Should check a AVXL balance of a Contract address Staking after unstake
- ✓ Should check a AVXL balance of a accounts[2]
- ✓ Should check Rewards stats of a accounts[2] after 51 days of staking and then unstake and claimed all
- ✓ Should not be able to claim tokens after stake of 51 days
- ✓ Should check unStake UserData
- ✓ Should check a AVXL balance of a account[3]
- ✓ Should check approval by accounts 3 to Staking contract to spend tokens on the behalf of staking
- ✓ Should Approve staking to spend specific tokens of accounts[3] (39ms)
- ✓ Should be able to transfer AVXL to accounts[3] (45ms)
- ✓ Should check a AVXL balance of a account[3]
- ✓ Should check stake contract user stats of accounts[3] before stake
- ✓ Should check total staked AVXL tokens using function
- ✓ Should check total avxl tokens staked
- ✓ Should check stake contract user stats
- ✓ Should Stake 200 LP tokens by account 4 (149ms)
- ✓ Should check unStake UserData before staking of accounts[2]
- ✓ Should not be able to claim tokens before stake by accounts[3] (39ms)

- ✓ Should not be able to unstake tokens before stake (41ms)
- ✓ Should not be able to stake less then 1 avxl token
- ✓ Should be able to stake when have avxl token from accounts 3 only 1 (51ms)
- ✓ Should check total staked AVXL tokens using function after staked by accounts[3]
- ✓ Should check total avxl tokens staked, after staked by accounts[3]
- ✓ Should check if stake holder or not , after staked by accounts[3]
- ✓ Should check stake contract user stats after staking by accounts[3]
- ✓ Should check a AVXL balance of a account[4]
- ✓ should check approval by accounts 4 to Staking contract to spend tokens on the behalf of staking
- ✓ should Approve staking to spend specific tokens of accounts[4] (43ms)
- ✓ Should be able to transfer AVXL to accounts[3] (56ms)
- ✓ Should check a AVXL balance of a account[4]
- ✓ Should check stake contract user stats of accounts[4] before stake
- ✓ Should check total staked AVXL tokens using function
- ✓ Should check total avxl tokens staked
- ✓ Should check stake contract user stats
- ✓ Should check unStake UserData before staking of accounts[4]
- ✓ Should not be able to claim tokens before stake by accounts[4] (75ms)
- ✓ Should not be able to unstake tokens before stake (85ms)
- ✓ Should not be able to stake less then 1 avxl token (44ms)
- ✓ Should be able to stake when have avxl token from accounts 3 only 1 (63ms)
- ✓ Should check total staked AVXL tokens using function after staked by accounts[3]
- ✓ Should check total avxl tokens staked, after staked by accounts[4]
- ✓ Should check if stake holder or not , after staked by accounts[4]
- ✓ Should check stake contract user stats after staking by accounts[4]
- ✓ Should be able to increase time to get 400 days
- ✓ Should check stake contract user stats

125 passing (10s)

0 Failed

AUTOMATED TESTS

Slither:

```
IERC20.transferFrom (StakingContract.sol#239-243) should be declared external
ERC20.increaseAllowance (StakingContract.sol#423-433) should be declared external
ERC20.decreaseAllowance (StakingContract.sol#449-462) should be declared external
Staking.stake (StakingContract.sol#703-733) should be declared external
Staking.unstake (StakingContract.sol#738-787) should be declared external
Staking.totalStakers (StakingContract.sol#807-809) should be declared external
Staking.afterUnstakeStats (StakingContract.sol#985-999) should be declared external
Staking.getUserStats (StakingContract.sol#1010-1043) should be declared external
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#public-function-that-could-be-declared-as-external
INFO:Detectors:
Detected issues with version pragma in StakingContract.sol:
- pragma solidity^0.5.0 (StakingContract.sol#3): it allows old versions
- pragma solidity^0.5.0 (StakingContract.sol#174): it allows old versions
- pragma solidity^0.5.0 (StakingContract.sol#266): it allows old versions
- pragma solidity^0.5.0 (StakingContract.sol#297): it allows old versions
- pragma solidity<=0.5.0<0.9.0 (StakingContract.sol#580): it has a complex pragma
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#incorrect-version-of-solidity
INFO:Detectors:
Function 'Context._msgSender' (StakingContract.sol#285-287) is not in mixedCase
Function 'Context._msgData' (StakingContract.sol#289-292) is not in mixedCase
Function 'ERC20._transfer' (StakingContract.sol#478-492) is not in mixedCase
Function 'ERC20._mint' (StakingContract.sol#503-509) is not in mixedCase
Function 'ERC20._burn' (StakingContract.sol#522-531) is not in mixedCase
Function 'ERC20._approve' (StakingContract.sol#546-556) is not in mixedCase
Function 'ERC20._burnFrom' (StakingContract.sol#564-574) is not in mixedCase
Parameter '_token' of Staking (StakingContract.sol#653) is not in mixedCase
Parameter '_pay' of Staking (StakingContract.sol#653) is not in mixedCase
Parameter '_address' of Staking.isStakeholder (StakingContract.sol#661) is not in mixedCase
Parameter '_address' of Staking.addStakeholder (StakingContract.sol#680) is not in mixedCase
Parameter '_address' of Staking.removeStakeholder (StakingContract.sol#691) is not in mixedCase
Parameter '_amount' of Staking.stake (StakingContract.sol#703) is not in mixedCase
Parameter '_address' of Staking.getBwardStats (StakingContract.sol#884) is not in mixedCase
Parameter '_address' of Staking.getUserStats (StakingContract.sol#1010) is not in mixedCase
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#conformance-to-solidity-naming-conventions
```

```
Staking.claimRewardAfterUnstake (StakingContract.sol#792-815) uses timestamp for comparisons
Dangerous comparisons:
- _weeksPassed > rewards[msg.sender].noOfWeeks (StakingContract.sol#802-804)
Staking.getRewardStats (StakingContract.sol#884-903) uses timestamp for comparisons
Dangerous comparisons:
- daysPassed > 30 (StakingContract.sol#949-959)
- _weeksPassed > rewards[_address].noOfWeeks (StakingContract.sol#911-913)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#block-timestamp
INFO:Detectors:
Different versions of Solidity is used in StakingContract.sol:
- Version used: ['>=0.5.0<0.9.0', 'ABIEncoderV2', '^0.5.0']
- StakingContract.sol#3 declares pragma solidity^0.5.0
- StakingContract.sol#174 declares pragma solidity^0.5.0
- StakingContract.sol#266 declares pragma solidity^0.5.0
- StakingContract.sol#297 declares pragma solidity^0.5.0
- StakingContract.sol#580 declares pragma solidity<=0.5.0<0.9.0
- StakingContract.sol#581 declares pragma experimentalABIEncoderV2
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#different-pragma-directives-are-used
INFO:Detectors:
ERC20.totalSupply (StakingContract.sol#335-337) should be declared external
IERC20.totalSupply (StakingContract.sol#184) should be declared external
IERC20.balanceOf (StakingContract.sol#189) should be declared external
ERC20.balanceOf (StakingContract.sol#342-344) should be declared external
IERC20.transfer (StakingContract.sol#198-200) should be declared external
ERC20.transfer (StakingContract.sol#354-357) should be declared external
ERC20.allowance (StakingContract.sol#362-368) should be declared external
IERC20.allowance (StakingContract.sol#209-212) should be declared external
ERC20.approve (StakingContract.sol#377-380) should be declared external
IERC20.approve (StakingContract.sol#228) should be declared external
ERC20.transferFrom (StakingContract.sol#394-409) should be declared external
IERC20.transferFrom (StakingContract.sol#239-243) should be declared external
ERC20.increaseAllowance (StakingContract.sol#423-433) should be declared external
ERC20.decreaseAllowance (StakingContract.sol#449-462) should be declared external
Staking.stake (StakingContract.sol#703-733) should be declared external
Staking.unstake (StakingContract.sol#738-787) should be declared external
```

```

INFO:Detectors:
Reentrancy in Staking.unstake (StakingContract.sol#738-787):
  External calls:
  - claimReward() (StakingContract.sol#756)
  State variables written after the call(s):
  - rewards (StakingContract.sol#776)
  - user (StakingContract.sol#779)
  - user (StakingContract.sol#780)
  - user (StakingContract.sol#781)
  - user (StakingContract.sol#782)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Staking.removeStakeholder (StakingContract.sol#691-697) does not use the value returned by external calls:
  - stakeholders.pop() (StakingContract.sol#695)
Staking.unstake (StakingContract.sol#738-787) does not use the value returned by external calls:
  - token.transfer(msg.sender, amount) (StakingContract.sol#785)
Staking.claimRewardAfterUnstake (StakingContract.sol#792-815) does not use the value returned by external calls:
  - token.transfer(msg.sender, amount) (StakingContract.sol#814)
Staking.claimReward (StakingContract.sol#820-862) does not use the value returned by external calls:
  - token.transfer(msg.sender, claimableRewards) (StakingContract.sol#858)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#unused-return
INFO:Detectors:
Reentrancy in Staking.unstake (StakingContract.sol#738-787):
  External calls:
  - claimReward() (StakingContract.sol#756)
  State variables written after the call(s):
  - totalStaked (StakingContract.sol#783)
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Staking.claimRewardAfterUnstake (StakingContract.sol#792-815) uses timestamp for comparisons:
  Dangerous comparisons:
  - _weeksPassed > rewards[msg.sender].noOfWeeks (StakingContract.sol#802-804)
Staking.getRewardStats (StakingContract.sol#884-903) uses timestamp for comparisons

```

Result:

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Surya:

```
ERC20 (Context, IERC20)
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #
- [Int] _transfer #
- [Int] _mint #
- [Int] _burn #
- [Int] _approve #
- [Int] _burnFrom #

Staking
- [Pub] <Constructor> #
- [Pub] isStakeholder
- [Int] addStakeholder #
- [Int] removeStakeholder #
- [Ext] stake #
- [Ext] unstake #
- [Int] claimRewardAfterUnstake #
- [Pub] claimReward #
- [Pub] totalStakers
- [Pub] getRewardStats
- [Pub] afterUnstakeStats
- [Pub] getUserStats

$) = payable function
= non-constant function
```

```

+ [Lib] SafeMath
- [Int] add
- [Int] sub
- [Int] sub
- [Int] mul
- [Int] div
- [Int] div
- [Int] mod
- [Int] mod

+ [Int] IERC20
- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] transfer #
- [Ext] allowance
- [Ext] approve #
- [Ext] transferFrom #

+ Context
- [Int] <Constructor> #
- [Int] _msgSender
- [Int] _msgData

+ ERC20 (Context, IERC20)
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance #
- [Pub] decreaseAllowance #
- [Int] _transfer #
- [Int] _mint #
- [Int] _burn #

```

Staking	Implementation			
L	<Constructor>	Public		NO
L	isStakeholder	Public		NO
L	addStakeholder	Internal		
L	removeStakeholder	Internal		
L	stake	External		NO
L	unstake	External		NO
L	claimRewardAfterUnstake	Internal		
L	claimReward	Public		NO
L	totalStakers	Public		NO
L	getRewardStats	Public		NO
L	afterUnstakeStats	Public		NO
L	getUserStats	Public		NO

Legend

Symbol	Meaning
	Function can modify state
	Function is payable







IERC20	Interface			
L	totalSupply	External		NO
L	balanceOf	External		NO
L	transfer	External		NO
L	allowance	External		NO
L	approve	External		NO
L	transferFrom	External		NO
Context	Implementation			
L	<CONSTRUCTOR>	Internal		
L	_msgSender	Internal		
L	_msgData	Internal		
ERC20	Implementation	Context, IERC20		
L	totalSupply	Public		NO
L	balanceOf	Public		NO
L	transfer	Public		NO
L	allowance	Public		NO

Sūrya's Description Report

Files Description Table

File Name	SHA-1 Hash
StakingContract.sol	ede8bcf30eb2fd9f0b4e9145d0d61659ae96a567

Contracts Description Table

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
L	add	Internal 		
L	sub	Internal 		
L	sub	Internal 		
L	mul	Internal 		
L	div	Internal 		
L	div	Internal 		
L	mod	Internal 		



CLOSING SUMMARY

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

Some low severity issues were detected; it is recommended to fix them.



DISCLAIMER

Cryptiocean audit is not a security warranty, investment advice, or an endorsement of the AvaXLauncher platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the AvaXLauncher Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.